

高等院校计算机科学与技术
“十五”规划教材

数字逻辑



●
武庆生
邓建
编著



机械工业出版社
CHINA MACHINE PRESS

高等院校计算机科学与技术“十五”规划教材

数 字 逻 辑

武庆生 邓 建 编著



机械工业出版社

“数字逻辑”是计算机专业本科学生的一门必修课程。它是“计算机组成原理”、“计算机接口技术”、“单片机及接口技术”等课程的先修课程之一。本课程的目的主要是使学生学习到从了解数字系统开始,直到能使用数字集成电路实现工程所需逻辑设计为止的完整过程。

本书根据《计算机学科教学计划》大纲编写,全书共8章,分别介绍了数字与编码、逻辑代数、集成门电路、组合逻辑电路、触发器、同步时序逻辑电路、异步时序逻辑电路、可编程逻辑电路等八个方面。

本书不仅介绍了经典的数字逻辑分析设计方法,而且还介绍了数字电路与逻辑设计的一些最新内容。全书体系新颖,取材科学,内容精炼,文字流畅,例题丰富。

本书可作为高等院校计算机、信息、电子工程、自动控制、通信等专业的教材,也可作为成人教育相关课程的教材,并可作为相关专业科技人员的参考书。

图书在版编目(CIP)数据

数字逻辑/武庆生,邓建编著. —北京:机械工业出版社,2004.4

(高等院校计算机科学与技术“十五”规划教材)

ISBN 7-111-13849-X

I. 数... II. ①武... ②邓... III. 数理逻辑—高等学校—教材 IV. 0141

中国版本图书馆CIP数据核字(2004)第002957号

机械工业出版社(北京市百万庄大街22号 邮政编码100037)

策 划:胡毓坚

责任编辑:陈振虹

责任印制:洪汉军

三河市宏达印刷有限公司印刷·新华书店北京发行所发行

2004年5月第1版·第1次印刷

787mm×1092mm $\frac{1}{16}$ ·14印张·346千字

0001—5000册

定价:20.00元

凡购本图书,如有缺页、倒页、脱页,由本社发行部调换

本社购书热线电话(010)68993821、88379646

封面无防伪标均为盗版

出版说明

信息技术高度普及的今天,具备一定层次的信息技术素养成为社会素质教育的一个重要目标,由此对高等院校的计算机专业教育提出了更高更新的要求。教育水平提高的关键是教学质量,那么对教学质量有直接影响的教材建设就成为了计算机专业教育的根本,为重中之重。

适逢高等院校计算机专业教育改革的关键时期,为配合相关的教材建设,机械工业出版社会同全国在该领域内享誉盛名、具备雄厚师资和技术力量的高等院校,包括清华大学、上海交通大学、南京大学、电子科技大学、东南大学、西安电子科技大学、解放军理工大学、北京科技大学等重点名校,组织长期从事教学工作的骨干教师,集思广益,对当前高等院校的教学现状开展了广泛而深入的研讨,继而紧密结合当前技术发展需要并针对教学改革所提出的问题,精心编写了这套面向普通高等院校计算机专业的系列教材,并陆续出版。

本套教材内容覆盖了普通高等院校计算机专业学生的必修课程,另外还恰如其分地添加了一些选修课程,总体上分为基础、软件、硬件、网络和多媒体五大类。在编写过程中,对教学改革力度比较大、内容新颖以及各院校急需的并且适应社会经济发展的新教材,优先选择出版。

本套教材注重系统性、普及性和实用性,力求达到专业基础课教材概念清晰、深度合理的标准,并且注意与专业课教学的衔接;专业课教材覆盖面广、深浅适中,在体现相关领域最新发展的同时注重理论联系实际。全套教材体现了教育改革的最新思想,可作为高等院校计算机科学与技术专业的教学用书,同时也是培训班和自学使用的最佳教材。

机械工业出版社

前 言

根据高等学校工科计算机专业“数字逻辑”课程教学大纲的要求，并考虑自控、信息、电子工程、通信等专业学习“数字逻辑”课程的需要，编者结合多年的教学经验编写了本书。

“数字逻辑”是计算机科学与技术（类）本、专科学生必修的一门重要专业基础课。本课程的目的是使学生从对数字系统的了解开始，到能使用数字集成电路实现工程所需的逻辑设计。这为数字计算机和其他数字系统的分析和设计奠定了良好的基础。熟练掌握数字系统逻辑分析和设计的方法，对从事计算机软硬件研制、开发和应用的工程技术人员是非常重要的。

数字集成电路是数字系统与计算机功能实现的基础，将数字逻辑设计和数字集成电路结合起来讲授学习，可使学生既掌握数字逻辑部件的分析与设计方法，又了解标准数字集成电路的原理和使用方法，并使学生了解数字集成器件的更新换代带来的数字系统设计方法的重大变化，从而跟上数字技术的快速发展。

全书共分 8 章，第 1 章为数制和码制，介绍数字系统中常用的数制及转换、码制和编码。第 2 章为逻辑代数，介绍逻辑代数、逻辑函数及函数化简。第 3 章为集成逻辑门，介绍典型 TTL 门，CMOS 门的结构和原理。第 4 章为组合逻辑电路，介绍组合逻辑电路的分析和设计方法以及典型中规模器件的原理和应用。第 5 章为触发器，介绍各种触发器的组成、原理和应用。第 6 章为同步时序逻辑电路，介绍同步时序逻辑电路的分析和设计方法以及中规模计数器的组成原理及应用。第 7 章为异步时序逻辑电路，介绍脉冲异步时序逻辑电路和电平异步时序逻辑电路的分析和设计方法以及中规模异步计数器的原理和应用。第 8 章为可编程逻辑电路，介绍了中大规模可编程逻辑器件的结构、原理和应用。

本书在内容和结构上力求主次分明、突出重点、叙述清楚、由浅入深。

本课程的先修课程是“电路与电子技术”。本课程的参考课时为 64 学时，使用者可根据需要和具体情况对内容进行取舍。

本书第 8 章所采用的逻辑符号部分是国外流行符号，其与中国国家标准的对照请参阅附录。

本书由武庆生和邓建编写。其中第 3、4、5、6 章由武庆生编写，第 1、2、7、8 章由邓建编写。在编写过程中得到了学院领导和同事的大力支持和关怀，陈雷霆副院长对本书给予了很多指导性的意见，傅彦副院长在百忙中也十分关心本书的编写工作。参加本书审校工作的还有黄俊翔、陈军等，谨对以上领导和同事的关心、支持、指导和帮助表示衷心的感谢。

由于编者水平有限，书中欠妥之处，敬请广大读者批评指正。

编 者

目 录

出版说明

前言

第 1 章 数制与码制	1
1.1 数制	1
1.1.1 进位计数制	1
1.1.2 二进制	2
1.1.3 任意进制转换为十进制	3
1.1.4 十进制转换为任意进制	3
1.2 带符号数的表示	4
1.2.1 原码及其运算	4
1.2.2 反码及其运算	5
1.2.3 补码及其运算	6
1.3 数的浮点表示	7
1.4 数和字符的编码	7
1.4.1 BCD 编码	8
1.4.2 格雷码	9
1.4.3 字符编码	9
1.4.4 奇偶校验码	10
1.5 习题	11
第 2 章 逻辑代数	12
2.1 基本概念	12
2.1.1 逻辑代数的定义	12
2.1.2 逻辑代数的基本运算	12
2.1.3 逻辑代数的复合运算	13
2.2 重要规则	13
2.3 逻辑函数的表达形式	14
2.4 逻辑函数的基本形式和标准形式	15
2.5 代数化简法	18
2.6 卡诺图化简法	20
2.6.1 卡诺图的结构	20
2.6.2 卡诺图的填入	20
2.6.3 卡诺图的性质	22
2.6.4 卡诺图化简逻辑函数	24
2.6.5 含有无关项的化简	26
2.7 习题	26

第3章 集成门电路	28
3.1 正逻辑和负逻辑	28
3.2 TTL 门电路	29
3.2.1 TTL 与非门	29
3.2.2 TTL 与非门的外特性	31
3.2.3 集电极开路输出(OC 门)和三态输出(TS 门)	32
3.3 CMOS 集成逻辑门电路	34
3.3.1 CMOS 反相器(非门)	35
3.3.2 CMOS 与非门	35
3.3.3 CMOS 或非门	35
3.3.4 CMOS 三态门	35
3.3.5 CMOS 漏极开路输出(OD 门)	36
3.3.6 CMOS 传输门	37
3.4 习题	37
第4章 组合逻辑电路	41
4.1 组合逻辑电路的分析	42
4.1.1 组合电路的分析步骤	42
4.1.2 组合电路的分析举例	42
4.2 组合逻辑电路设计	44
4.2.1 设计步骤	44
4.2.2 设计举例	44
4.3 加法器	48
4.3.1 半加器和全加器	49
4.3.2 加法器模块	50
4.3.3 加法器的应用	51
4.4 数值比较器	54
4.4.1 一位数值比较器	54
4.4.2 四位数值比较器	54
4.4.3 集成比较器的应用	55
4.5 编码器和译码器	57
4.5.1 编码器	57
4.5.2 译码器	60
4.6 数据选择器和数据分配器	70
4.6.1 数据选择器	70
4.6.2 数据分配器	74
4.7 组合逻辑电路中的竞争与冒险	75
4.7.1 竞争和冒险现象	75
4.7.2 怎样判定电路中有无险象	76
4.7.3 险象的消除和减弱	77

4.8	习题	78
第 5 章	触发器	81
5.1	基本 RS 触发器	81
5.1.1	用与非门构成的基本 RS 触发器	81
5.1.2	用或非门构成的基本 RS 触发器	83
5.2	钟控触发器	85
5.2.1	钟控 RS 触发器	85
5.2.2	钟控(电平型) D 触发器	86
5.3	主从触发器	86
5.3.1	主从 RS 触发器	86
5.3.2	主从 JK 触发器	88
5.4	边沿触发器	90
5.4.1	维持-阻塞 D 触发器	90
5.4.2	边沿 JK 触发器	91
5.5	集成触发器	92
5.5.1	集成 D 触发器	92
5.5.2	集成 JK 触发器	93
5.6	其他功能的触发器	94
5.6.1	T 触发器	94
5.6.2	T' 触发器(翻转触发器)	94
5.7	各类触发器的相互转换	94
5.7.1	JK 触发器转换为 D、T、T' 和 RS 触发器	94
5.7.2	D 触发器转换为 JK、T、T' 和 RS 触发器	96
5.8	集成触发器的参数	97
5.8.1	触发器的静态参数	97
5.8.2	触发器的动态参数	97
5.9	习题	98
第 6 章	同步时序逻辑电路	102
6.1	时序逻辑电路的结构和类型	102
6.1.1	时序逻辑电路的结构和特点	102
6.1.2	时序逻辑电路分类	103
6.2	同步时序逻辑电路的分析	104
6.2.1	分析步骤	104
6.2.2	分析举例	104
6.3	同步时序逻辑电路的设计	111
6.3.1	设计步骤	111
6.3.2	建立原始状态图(或状态表)	112
6.3.3	状态化简	116
6.3.4	状态分配	123

6.3.5	同步时序电路设计举例	125
6.4	计数器及其应用	134
6.4.1	计数器的特点和分类	134
6.4.2	n 位二进制计数器	134
6.4.3	十进制计数器	139
6.4.4	利用反馈归 0 法和反馈置数法构成任意进制计数器	144
6.4.5	计数器容量的扩展	145
6.5	寄存器	147
6.5.1	锁存器	147
6.5.2	基本寄存器	147
6.5.3	移位寄存器	148
6.5.4	移位寄存器型计数器	149
6.6	习题	153
第 7 章	异步时序逻辑电路	157
7.1	脉冲异步时序逻辑电路	157
7.1.1	脉冲异步时序逻辑电路的分析	158
7.1.2	脉冲异步时序逻辑电路的设计	161
7.2	电平异步时序逻辑电路	165
7.2.1	电平异步时序逻辑电路的分析	167
7.2.2	电平异步时序逻辑电路中的竞争与险象	169
7.2.3	电平异步时序逻辑电路设计	171
7.3	习题	175
第 8 章	可编程逻辑电路	177
8.1	概述	177
8.2	只读存储器	179
8.3	可编程逻辑阵列 PLA	182
8.4	可编程阵列逻辑 PAL	185
8.5	通用阵列逻辑 GAL	190
8.6	现场可编程门阵列 FPGA	204
8.7	复合可编程逻辑器件 CPLD	209
8.8	习题	212
附录	逻辑符号对照表	215
参考文献		216

第1章 数制与码制

21 世纪人类已经进入信息时代, 信息技术与国民经济、人们的日常生活息息相关, 围绕信息的采集、存储、传输、计算、利用等技术已成为当前科技研究的重要内容。信息技术处理的信号分为模拟信号和数字信号两大类, 如图 1-1 所示。在时间和数值上是连续的信号称为模拟信号。而数字信号是指在低电平和高电平两种状态(即 0 和 1 两个逻辑值)之间作阶跃式变化的信号, 这种在时间上和数值上不连续的信号又称为离散信号。

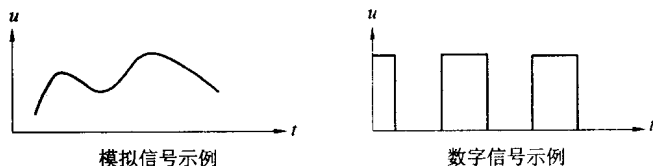


图 1-1 模拟信号与数字信号

对数字信号进行传递、处理的电路称为数字电路。由于数字电路不仅能对信号进行数值运算, 而且还有逻辑运算和逻辑判断的功能, 数字电路的输入量和输出量之间的关系是一种因果关系, 它可以用逻辑函数来描述, 所以又称为数字逻辑电路或逻辑电路。数字逻辑电路主要研究电路输出信号状态与输入信号状态之间的逻辑关系。

在数字系统中, 目前广泛采用二进制系统。

1.1 数制

数制即计数体制, 它是按照一定规律表示数值大小的计数方法。日常生活中最常用的计数体制是十进制, 数字电路中最常用的计数体制是二进制。

在数字电路中, 常用一定位数的二进制数码表示不同的事物或信息, 这些数码称为代码, 编制代码时要遵循一定的规则, 这些规则叫码制。

1.1.1 进位计数制

人类在长期的生产实践中学会了用 10 个指头(0~9)计数, 很显然仅用一位数码往往不够, 必须采用多位数码按先后次序把它们排成数位, 由低到高进行计数, 计满后进位, 这就产生了我们最熟悉的十进制。数位计数制(positional number systems)是人们对数量计数的规律的总结。任何一种进位计数制都包含着基数(base/radix)和位权(weight)两个特征。

1. 基数

基数是指数制中所采用的数字符号个数, 基数为 R 的数制称为 R 进制。 R 进制中能表示 $0 \sim R-1$ 的共 R 个数字符号。

2. 位权

位权(简称为权)是指进位计数制中不同数位上的数值, 即平常所讲的个位、十位、百

位或十分位、百分位等。

进位规律是“逢 R 进一”。一个 R 进制数 N 可表示为

$$(N)_R = (K_{n-1} K_{n-2} \cdots K_1 K_0 \cdot K_{-1} \cdots K_{-m})_R \quad \text{并列表示法 (位置记数法)}$$

$$\text{或 } (N)_R = \sum_{i=-m}^{n-1} K_i R^i \quad \text{多项式表示法 (按权展开式)}$$

式中 R 为基数, K_i 为 $0 \sim R-1$ 中的任何一个数字符号, n 为整数部分位数, m 为小数部分位数, R^i 为第 i 位的位权。

$$\text{如: } (6666)_{10} = 6 \times 10^3 + 6 \times 10^2 + 6 \times 10^1 + 6 \times 10^0$$

$$(4032.8)_{10} = 4 \times 10^3 + 0 \times 10^2 + 3 \times 10^1 + 2 \times 10^0 + 8 \times 10^{-1}$$

1.1.2 二进制

日常生活中常采用十进制, 所以在早期的计算工具的设计中也采用十进制, 但随着数字系统的发展, 为了便于工程实现, 广泛采用二进制。这是因为二进制表示的数的每一位只取数码 0 或 1, 可以用具有两种不同状态的电子元件来表示。它的运算规则简单, 0、1 与逻辑命题中的真假相对应, 为计算机中实现逻辑运算和逻辑判断提供了有利条件。

二进制的基数是 2, 其计数规律是逢二进一。一个二进制数可以表示成

$$(N)_2 = (K_{n-1} K_{n-2} \cdots K_1 K_0 \cdot K_{-1} \cdots K_{-m})_2 \quad \text{并列表示法 (位置记数法)}$$

$$\text{或 } (N)_2 = \sum_{i=-m}^{n-1} K_i 2^i \quad \text{多项式表示法 (按权展开式)}$$

$$\begin{aligned} \text{如: } (101.001)_2 &= 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\ &= 1 \times 4 + 0 \times 2 + 1 \times 1 + 0 \times 0.5 + 0 \times 0.25 + 1 \times 0.125 \\ &= (5.125)_{10} \end{aligned}$$

采用二进制计数制, 对数字的运算、存储和传输极为方便, 然而二进制数书写起来很不方便。为此人们经常采用八进制数和十六进制来进行书写或打印。

八进制数的基是 8, 它有 0~7 共 8 个符号。十六进制数的基是 16, 它有 16 个符号, 除了 0~9 以外, 还需补充 6 个符号, 它们是 A (代表 10), B (代表 11), C (代表 12), D (代表 13), E (代表 14), F (代表 15)。

由于 $2^3=8$, 所以 1 位八进制数所能表示的数值, 正好和 3 位二进制数对应。同理, 由于 $2^4=16$, 所以 1 位十六进制数所能表示的数值, 正好和 4 位二进制数对应。

根据上述特点, 二进制、八进制、十六进制之间可以很方便地相互转换。二进制数转换为八进制数 (十六进制) 时以小数点为界, 分别往左、右每 3 位 (4 位) 为一组, 最后不足 3 位 (4 位) 时用 0 补充, 然后写出每组对应的八进制 (十六进制) 字符, 即为对应八进制数 (十六进制)。

【例 1-1】 将 $(374.27)_8$ 和 $(AF4.74)_{16}$ 转为二进制。

$$\text{解: } (374.27)_8 = 011\ 111\ 100.010\ 111$$

$$(AF4.74)_{16} = 1010\ 1111\ 0100.0111\ 0100$$

【例 1-2】 将 $(10110.1011)_2$ 转为八进制、十六进制数。

$$\begin{aligned}\text{解: } (10110.1011)_2 &= 010\ 110.101\ 100 \\ &= (2\ 6.\ 5\ 4)_8 \\ (10110.1011)_2 &= 0001\ 0110.1011 \\ &= (1.\ 6.\ B)_{16} \\ \text{即 } (10110.1011)_2 &= (26.54)_8 = (16.B)_{16}\end{aligned}$$

1.1.3 任意进制转换为十进制

将一个任意进制数转换成十进制数时采用多项式替代法, 即将 R 进制数按权展开, 求出各位数值之和, 即可得到相应的十进制数。

【例 1-3】 将 $(10110.1011)_2$ 、 $(4365.5)_8$ 、 $(1C4)_{16}$ 、 $(D8.A)_{16}$ 转换成十进制数。

$$\begin{aligned}\text{解: } (10110.1011)_2 &= 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\ &= 16 + 0 + 4 + 2 + 0 + 0.5 + 0 + 0.125 \\ &= (22.625)_{10} \\ (436.5)_8 &= 4 \times 8^2 + 3 \times 8^1 + 6 \times 8^0 + 5 \times 8^{-1} = (286.625)_{10} \\ (1C4)_{16} &= 1 \times 16^2 + C \times 16^1 + 4 \times 16^0 = 256 + 192 + 4 = (452)_{10} \\ (D8.A)_{16} &= 13 \times 16^1 + 8 \times 16^0 + 10 \times 16^{-1} = (216.625)_{10}\end{aligned}$$

1.1.4 十进制转换为任意进制

十进制数转换为其他进制数时, 整数部分和小数部分要分别转换。整数部分采用除基取余法, 小数部分采用乘基取整法。

除基取余法是将十进制整数 N 除以基数 R , 取余数为 K_0 , 再将所得商除以 R , 取余数为 K_1 , 依此类推直至商为 0, 取余数为 K_{n-1} 为止, 即得到与 N 对应的 R 进制数 $(K_{n-1} \cdots K_1 K_0)_R$ 。

【例 1-4】 将 $(217)_{10}$ 转换成二进制数。

$$\begin{array}{r|l} \text{解: } 2 & 217 \\ \hline 2 & 108 \\ \hline 2 & 54 \\ \hline 2 & 27 \\ \hline 2 & 13 \\ \hline 2 & 6 \\ \hline 2 & 3 \\ \hline 2 & 1 \\ \hline & 0 \end{array} \quad \begin{array}{l} \text{余数} \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{array} \quad \begin{array}{l} \text{最低位 } K_0 \\ \\ \\ \\ \\ \\ \\ \text{最高位 } K_7 \end{array}$$

$$\text{所以 } (217)_{10} = (11011001)_2$$

乘基取整法是将十进制小数 N 乘以基数 R , 取整数部分为 K_1 , 再将其小数部分乘以 R , 取整数部分为 K_2 , 依此类推直至其小数部分为 0 或达到规定精度要求, 取整数部分记为 K_m 为止, 即可得到与 N 对应的 m 位 R 进制数 $(0.K_1 K_2 \cdots K_m)_R$ 。

【例 1-5】 将十进制小数 0.6875 转换成二进制小数。

解: 0.6875

$\times \quad 2$	整数部分	
1	$.3750$	1 最高位 K_1
$\times \quad 2$	0	$.7500$ 0 K_2
$\times \quad 2$	1	$.5000$ 1 K_3
$\times \quad 2$	1	$.0000$ 1 最低位 K_4

所以 $(0.6875)_{10} = (0.1011)_2$

若十进制小数不能用有限位二进制小数精确表示, 则应根据精度要求, 当要求二进制数取 m 位小数时可求出 $m+1$ 位, 然后对最低位作 0 舍 1 入处理。

【例 1-6】 将十进制小数 0.324 转换成二进制小数, 保留 3 位小数。

解: 0.324

$\times \quad 2$	整数部分	
0	$.648$	0 最高位 K_1
$\times \quad 2$	1	$.296$ 1 K_2
$\times \quad 2$	0	$.592$ 0 K_3
$\times \quad 2$	1	$.184$ 1 最低位 K_4

所以 $(0.324)_{10} \approx (0.0101)_2 \approx (0.011)_2$

如果一个十进制数既有整数又有小数, 可将整数和小数分别进行转换, 然后合并就可得到结果。

【例 1-7】 将 $(24.625)_{10}$ 转换成二进制。

解: $(24.625)_{10} = (24)_{10} + (0.625)_{10}$
 $= (11000)_2 + (0.101)_2$
 $= (11000.101)_2$

1.2 带符号数的表示

1.2.1 原码及其运算

日常书写时在数值前面用“+”号表示正数, “-”号表示负数, 这种带符号二进制数称为真值。在计算机处理时, 必须将“+”和“-”转换为数码, 符号数码化的数称为机器数。一般将符号位放在最高位, 用“0”表示符号为“+”, 用“1”表示符号为“-”。根据数值位的表示方法不同, 有三种类型: 原码、反码和补码。

原码是指符号位用 0 表示正, 1 表示负; 数值位与真值一样, 保持不变。

如: $X_1 = +1101$, $X_2 = -1101$, 则 $[X_1]_{\text{原}} = 01101$, $[X_2]_{\text{原}} = 11101$ 。

$X_3 = +0.1011$, $X_4 = -0.1011$, 则 $[X_3]_{\text{原}} = 0.1011$, $[X_4]_{\text{原}} = 1.1011$ 。

可以用下面的公式来将真值转换为原码（含 1 位符号位，共 n 位）。

整数：

$$[N]_{\text{原}} = \begin{cases} N & 0 \leq N < 2^{n-1} \\ 2^{n-1} - N & -2^{n-1} < N \leq 0 \end{cases}$$

表示范围：-127 ~ +127（8 位整数）

注意整数“0”的原码有两种形式，即 $[+0]_{\text{原}} = 00 \cdots 0$ 和 $[-0]_{\text{原}} = 10 \cdots 0$ 。

纯小数：

$$[N]_{\text{原}} = \begin{cases} N & 0 \leq N < 1 \\ 1 - N & -1 < N \leq 0 \end{cases}$$

纯小数“0.0”的原码也有两种形式。

原码的优点是容易理解，它和代数中的正负数的表示方法很接近。但原码的缺点是“0”的表示有两种；原码的运算规则复杂，若 $X=Y+Z$ ，但 $(X)_{\text{原}} \neq (Y)_{\text{原}} + (Z)_{\text{原}}$ 。

例如 $(+1000)_2 = (+1011)_2 + (-0011)_2$ ，但是 $(+1011)_{\text{原}} = 01011$ ， $(-0011)_{\text{原}} = 10011$ ，它们直接相加不等于 $(+1000)_2$ 的原码 (01000) ，所以为了使结果正确，原码的加法规则为：

（1）判断被加数和加数的符号是同号还是异号。

（2）同号时，做加法，结果的符号就是被加数的符号。

（3）异号时，先比较被加数和加数的数值（绝对值）的大小，然后由大值减去小值，结果的符号取大值的符号。

由于原码的运算规则复杂，为了简化机器数的运算，因此需要寻找其他表示负数的方法。

1.2.2 反码及其运算

反码的符号位用 0 表示正，1 表示负；正数反码的数值位和真值的数值位相同，而负数反码的数值位是真值的按位变反。

可以用下面的公式来将真值转换为反码（含 1 位符号位，共 n 位）。

整数：

$$[N]_{\text{反}} = \begin{cases} N & 0 \leq N < 2^{n-1} \\ (2^n - 1) + N & -2^{n-1} < N \leq 0 \end{cases}$$

表示范围：-127 ~ +127（8 位整数）

注意整数“0”的反码有两种形式，即 $[+0]_{\text{反}} = 00 \cdots 0$ 和 $[-0]_{\text{反}} = 11 \cdots 1$ 。

纯小数：

$$[N]_{\text{反}} = \begin{cases} N & 0 \leq N < 1 \\ (2 - 2^{-m}) + N & -1 < N \leq 0 \end{cases}$$

同理，纯小数“0.0”的反码也有两种形式。

采用反码进行加、减运算时，无论进行两数相加还是两数相减，均可通过加法实现。

$$[X1+X2]_{反} = [X1]_{反} + [X2]_{反}$$

$$[X1-X2]_{反} = [X1]_{反} + [-X2]_{反}$$

运算时符号位和数值位一起参加运算，当符号位有进位产生时，应将进位加到运算结果的最低位，才能得到最后正确结果。我们称之为“循环相加”（或“循环进位”）。

【例 1-8】 已知 $X1 = +0.1110$, $X2 = +0.0101$, 求 $X1-X2$ 。

解: $X1-X2$ 可通过反码相加实现。运算如下:

$$\begin{aligned} [X1-X2]_{反} &= [X1]_{反} + [-X2]_{反} = 0.1110 + \\ &\quad 1.1010 \\ &= 10.1000 \quad (\text{符号位上有进位}) \\ &= 0.1001 \end{aligned}$$

在反码中, $[X]_{反} \rightarrow [-X]_{反}$ 的方法是符号位连同数值位一起 0 变 1, 1 变 0。

如: $[X]_{反}$ 为 0.0101, 则 $[-X]_{反}$ 为 1.1010。

1.2.3 补码及其运算

补码是符号位用 0 表示正, 1 表示负; 正数补码的数值位和真值相同, 而负数补码的数值位是真值的按位变反, 最低位加 1。

如: $X1 = +1101$, $X2 = -1101$, 则 $[X1]_{补} = 01101$, $[X2]_{补} = 10011$ 。

$X3 = +0.1011$, $X4 = -0.1011$, 则 $[X3]_{补} = 0.1011$, $[X4]_{补} = 1.0101$ 。

可以用下面的公式来将真值转换为补码 (含 1 位符号位, 共 n 位)。

整数:

$$[N]_{补} = \begin{cases} N & 0 \leq N < 2^{n-1} \\ 2^n + N & -2^{n-1} \leq N < 0 \end{cases}$$

表示范围: $-128 \sim +127$ (8 位整数), 补码 10000000 表示 -128 (-2^7)。

注意整数“0”的补码只有一种形式, 即 00...0。

纯小数:

$$[N]_{补} = \begin{cases} N & 0 \leq N < 1 \\ 2 + N & -1 \leq N < 0 \end{cases}$$

采用补码进行加、减运算时, 无论进行两数相加还是两数相减, 均可通过加法实现。

$$[X1+X2]_{补} = [X1]_{补} + [X2]_{补}$$

$$[X1-X2]_{补} = [X1]_{补} + [-X2]_{补}$$

运算时符号位和数值位一起参加运算, 不必处理符号位上的进位。

【例 1-9】 已知 $X1 = +0.1110$, $X2 = +0.0101$, 求 $X1-X2$ 。

解: $X1-X2$ 可通过补码相加实现。运算如下:

$$\begin{aligned} [X1-X2]_{补} &= [X1]_{补} + [-X2]_{补} = 0.1110 + \\ &\quad 1.1011 \\ &= 10.1001 \quad (\text{丢弃符号位上的进位}) \\ &= 0.1001 \end{aligned}$$

当真值用补码表示时，补码加法的规律和无符号数的加法规律完全一样，因此简化了加法器的设计。

从 $[X]_{\text{补}}$ 求 $[-X]_{\text{补}}$ 的方法是符号位连同数值位一起变反，尾数再加 1。

【例 1-10】 已知 $X = +100\ 1001$ ，求 $[X]_{\text{补}}$ 和 $[-X]_{\text{补}}$ 。

解： $[X]_{\text{补}} = 0100\ 1001$ ， $[-X]_{\text{补}} = 1011\ 0110 + 1 = 1011\ 0111$

1.3 数的浮点表示

当一个数既有整数部分，又有小数部分时，如何在机器里表达呢？我们知道，可以将 353.75 表示为 0.35375×10^3 的形式，所以在机器里可以将二进制数表示为这种浮点数的形式。其一般形式为 $N = 2^J \times S$ ，其中 2^J 为 N 的指数部分，表示小数点的位置， J 称为阶码， S 为 N 的尾数部分，表示数的符号和有效数字。阶码的符号位称为阶符 J_f ，尾数的符号位称为尾符 S_f 。

如： $N = -0.00001101_2 = 2^{(-4)}_{10} \times (-0.1101) = 2^{-100} \times (-0.1101)$

规格化数是使尾数最高数值位非 0，可以提高运算精度。例如

$$1011 \rightarrow 2^{101} \times 0.01011 \rightarrow 2^{100} \times 0.1011$$

如果尾数的数值部分只有 4 位，则前一种表示将产生误差。在规格化数中，当用原码表示尾数时，使小数点后的最高数据位为 1；当用补码表示尾数时，使小数点后的数值最高位与数的符号位相反。

两个浮点数作加减运算时要要先对阶。

【例 1-11】 已知 $N_1 = 2^{011} \times 0.1001$ ， $N_2 = 2^{001} \times 0.1100$ ，求 $N_1 + N_2$ 。

解：先对阶： $N_2 = 2^{001} \times 0.1100 = 2^{011} \times 0.0011$ （小数点左移 2 位，阶码加 2）

$$\begin{aligned} N_1 + N_2 &= 2^{011} \times 0.1001 + 2^{011} \times 0.0011 \\ &= 2^{011} (0.1001 + 0.0011) \\ &= 2^{011} \times 0.1100 \end{aligned}$$

两个浮点数作乘法，其规则为：

若 $N_1 = 2^{j_1} \times S_1$ ， $N_2 = 2^{j_2} \times S_2$ 则

$$\begin{aligned} N_1 \times N_2 &= (2^{j_1} \times S_1) \times (2^{j_2} \times S_2) \\ &= 2^{(j_1 + j_2)} \times (S_1 \times S_2) \\ N_1 / N_2 &= 2^{(j_1 - j_2)} \times (S_1 / S_2) \end{aligned}$$

1.4 数和字符的编码

数字系统中的信息有两类：一类是数码信息，另一类是代码信息。数码信息的表示方法如前所述，以便在数字系统中进行运算、存储和传输。为了表示字符等一类被处理的信息，也需要用一定位数的二进制数码表示，这个特定的二进制码称为代码。注意，“代码”和“数码”的含义不尽相同，代码是不同信息的代号。

每个信息制定一个具体的码字去代表它，这一指定过程称为编码。由于指定的方法不是惟一的，故对一组信息存在着多种编码方案。

1.4.1 BCD 编码

在数字系统中, 各种数据要转换为二进制代码才能进行处理, 而人们习惯于使用十进制数, 所以在数字系统的输入输出中仍采用十进制数, 这样就产生了用四位二进制数表示一位十进制数的方法, 这种用于表示十进制数的二进制代码称为二—十进制代码 (Binary Coded Decimal), 简称为 BCD 码。它具有二进制数的形式以满足数字系统的要求, 又具有十进制的特点 (只有 10 种有效状态)。在某些情况下, 计算机也可以对这种形式的数直接进行运算。常见的 BCD 码表示有 8421 码、2421 码和余 3 码。

余 3 码是每个 8421 码加 3。三种 BCD 码都是用四位二进制代码表示一位十进制数字, 与十进制数之间的转换是以四位二进制对应一位十进制直接进行变换。一个 n 位十进制数对应的 BCD 码一定为 $4n$ 位。

在 BCD 编码中, 若四位二进制一组中的每位都有固定权值, 则称为有权码 (weighted code), 如 8421 码、2421 码是有权码。余 3 码是无权码。

当两个十进制数为互反, 它们对应的二进制码互反, 则称为自补码 (self-complementing code)。2421 码、余 3 码是自补码。

常见 BCD 编码如表 1-1 所示。

表 1-1 常用 BCD 码

十 进 制 数	8421 码	2421 码	余 3 码
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

8421 码是一种使用最广泛的 BCD 码, 是一种有权码, 其各位的权分别是 (从最有效高位开始到最低有效位) 8,4,2,1。

【例 1-12】 写出十进数 $(1592)_{10}$ 对应的 8421 码和余 3 码。

解: $(1592)_{10}=(0001\ 0101\ 1001\ 0010)_{8421}$
 $= (0100\ 1000\ 1100\ 0101)_{\text{余 3 码}}$

【例 1-13】 写出 8421 码 $(1101001.01011)_{8421}$ 对应的十进制数。

解: $(1101001.01011)_{8421}=0110\ 1001.0101\ 1000_{8421}=69.58$

在使用 8421BCD 码时一定要注意到其有效的编码仅 10 个, 即: 0000~1001。四位二进制数的其余六个编码 1010,1011,1100,1101,1110,1111 不是有效编码。在余 3 码中 0000~0010, 1101~1111 这 6 个编码不是有效编码。在 2421 码中, 0101~1010 这 6 个编码仍表示有效的十进制数, 只不过由于它们和已有的十进制数重复, 所以不使用。